Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 189 (2008) 652–668
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A branch-and-price approach for integrating nurse and surgery scheduling

Jeroen Beliën *, Erik Demeulemeester

Decision Sciences and Information Management, Faculty of Economics and Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, 3000 Leuven, Belgium

Received 12 October 2005; accepted 6 October 2006
Available online 18 January 2007

Abstract

A common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. The operating room is considered to be the main engine and hence the main generator of variance in the hospital. The purpose of this paper is threefold. First of all, we present a concrete model that integrates both the nurse and the operating room scheduling process. Second, we show how the column generation technique approach, one of the most employed exact methods for solving nurse scheduling problems, can easily cope with this model extension. Third, by means of a large number of computational experiments we provide an idea of the cost saving opportunities and required solution times.

Keywords: Scheduling; Integer programming; Branch-and-price; OR in health services

1. Introduction

During the last decades, cost pressures on hospitals have increased dramatically. This emphasis on cost containment has forced hospital executives to run their organizations in a more business-like manner. The constant challenge is to provide high-quality service at ever reduced costs. In order to achieve this purpose, inefficient use of resources should be identified and actions should be taken to eliminate these sources of waste. Operations research techniques are increasingly being used to assist in this complicated task.

As nursing services account for an important part of a hospital's annual operating budget, concentrating on this resource can lead to substantial savings. The situation is exacerbated by an acute shortage of nurses in all western countries, said to be 120,000 today and expected to grow to 808,000 by 2020 in the United States alone (USDHHS, 2002). Hence, it is of vital importance that nurses are used as much as possible at the right time and at the right place. This goal is hard to achieve because of two reasons. The first one is inherent in service organizations for which human resources outnumber all other types of resources. Unlike machines, staff schedules are restricted by collective agreement requirements. These form an important hindrance for the flexibility with which nurses are scheduled.

* Corresponding author.
E-mail addresses: jeroen.belien@econ.kuleuven.be (J. Beliën), erik.demeulemeester@econ.kuleuven.be (E. Demeulemeester).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.10.060

A second reason is the presence of variability. One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. On days when the workload is too high, the quality of care decreases because it is too costly to staff for peak loads. On days when the workload is too low, there is waste. Fortunately, the situation is not as chaotic as it seems to be at first sight. As pointed out by Litvak and Long (2000), an important amount of the variability can effectively be managed and reduced by a thorough analysis of the existing system and by appropriate decision taking. Special emphasis is put on the operating room since it is considered to be the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operating room schedule process into the nurse scheduling process is a simple yet effective way to achieve considerable savings in staffing costs.

This paper is organized as follows. In Section 2, a discussion of the background together with a brief literature review is given. In Section 3, a general overview of the model together with a branch-and-price solution approach is presented. Section 4 provides more details on both pricing problems, while a general overview of the branch-and-price algorithm is given in Section 5. In Section 6, some computational issues are discussed and in Section 7, extensive computational results are given. Finally, Section 8 draws conclusions and lists some topics for further research.

2. Background and literature review

Nurse scheduling problems are frequently encountered in the operations research literature. Excellent surveys on medical staff rostering problems can be found in Cheang et al. (2003) and Burke et al. (2004). Several studies in the literature have utilized mathematical programming techniques to assist in finding efficient staff schedules (see, e.g., Miller et al., 1976; Warner, 1976; Azaiez and Al Sharif, 2005; Beaumont, 1997; Bard et al., 2003). These problems typically involve some kind of set covering or set partitioning formulation. The main drawback, however, is that these models can have far more variables than can be reasonably attacked directly. Therefore, the linear program (LP) is often solved using column generation (see, e.g., Caprara et al., 2003; Jaumard et al., 1998; Bard and Purnomo, 2005a,b; Mehrotra et al., 2000; Mason and Smith, 1998).

Other successful approaches to deal with real-life nurse rostering problems include constraint programming and heuristic procedures. Constraint programming models have been proposed by Okada and Okada (1988), Darmoni et al. (1995), Weil et al. (1995), Cheng et al. (1997), Abdennadher and Schlenker (1999), Meyer auf'm Hofe (2001) and Musliu et al. (2000). Many modern attempts to solve complex scheduling problems involve tabu search techniques (Dowsland, 1998; Burke et al., 1999, 2004, 2006; De Causmaecker and Vanden Berghe, 2003) and genetic and memetic algorithms (Burke et al., 2001; Ozcan, 2005).

The surgery scheduling part in this paper uses integer programming to develop a master surgery schedule. Examples of integer programming for operating room scheduling exist in the literature, but mainly for case mix planning decisions; i.e., determining *how many* operating room time each surgeon obtains, but not *when* the surgeons get operating room time (e.g., Hughes and Soliman, 1985; Rifai and Pecenka, 1989; Robbins and Tuntiwongbiboon, 1989; Blake and Carter, 2002, 2003). Blake et al. (2002) propose an integer programming model for building a cyclic schedule that allocates to each surgical group a number of operating room hours as close as possible to its target operating room hours, while at the same time keeping the schedule as simple (repetitive) as possible (see also Blake and Donald, 2002). Beliën and Demeulemeester (2006) propose a number of integer programming models for building surgery schedules with leveled resulting bed occupancy. A leveled bed occupancy leads in many cases to a leveled workload pattern which on its turn usually leads to a favorable workload pattern for nurses. However, since the scheduling of nurses by itself already entails a lot of constraints, one cannot really judge the quality of a surgery schedule with respect to the resulting workload distribution, without explicitly taking these constraints into account. Hence, an integrated approach of both scheduling fields is required in order to build a high-quality surgery schedule that aims at a reduction in the staffing costs. Also heuristic procedures have been used. For instance, Hans et al. (2005) propose several constructive and local search heuristics for the robust surgery loading problem. On a more detailed level, patients have to be assigned to nurses. Mullinax and Lawley (2002) have developed a mathematical programming approach and heuristic for this assignment problem that aims at a balanced work-

load. Punnakitikashem et al. (2005) propose a stochastic programming approach that takes into account uncertainty.

The methodology presented in this paper has some similarities with models for integrating the scheduling of project tasks and employees (Alfares and Bailey, 1997; Alfares et al., 1999). Although several authors mention the influence of the surgery scheduling process on the nurses' workload (e.g., Litvak and Long, 2000; Jun et al., 1999), as far as we know, no models have been proposed to integrate both areas of decision making.

3. Model description

3.1. Schematic overview

Fig. 1 contains a schematic overview of the general idea outlined in this paper. The input for the nurse scheduling process (at the right) consists of the restrictions implied on the individual nurse roster lines on the one hand and the workload distribution over time on the other hand. The workload distribution itself is determined by the master surgery schedule. In order to be able to deduce the workload from the surgery schedule one also has to know the workload contributions of each specific type of surgery. The dotted arrow at the bottom indicates the feedback that could be given from the nurse scheduling process to the surgery scheduling process in order to produce more favorable surgery schedules with respect to the resulting workloads. However, the freedom in modifying the surgery schedule is limited, since the master surgery schedule itself is restricted by a set of specific surgery constraints (e.g., capacity and demand constraints).

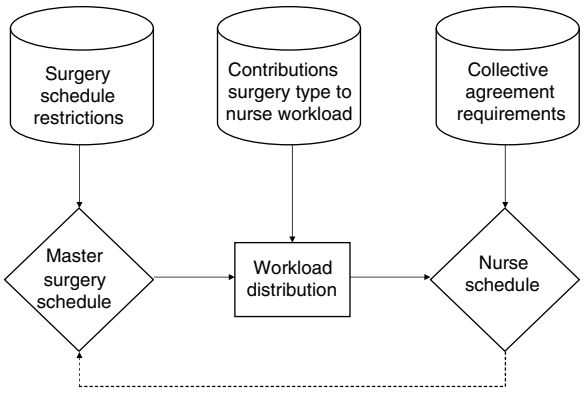

Fig. 1. Schematic overview of the general idea.

In what follows we will describe a mathematical model for implementing this idea. We focus on the minimization of the total required number of nurses. The reason for this objective is that it allows for a quantitative measure of the resulting benefits, i.e., the decrease in staffing cost. Obviously, this quantitative benefit can easily be turned into a qualitative benefit by employing the saved nurse(s) on moments when they are most needed. The computational complexity of combining surgery scheduling and nurse rostering is, however, not the main obstacle for an integrated approach. The current practice shows us that nurse rostering is subordinate to surgery scheduling in many hospitals; i.e., surgery schedules are built first and the result is taken into account as a preset constraint in a nurse rostering problem instance. Hence, in order to implement the proposed model, a structural change in the way of organization, in the mentality of the surgeons will be required.

3.2. The nurse scheduling problem

The nurse scheduling problem (NSP) consists of generating a configuration of individual schedules over a given time horizon. The configuration of nurse schedules is generated so as to fulfill collective agreement requirements and the hospital staffing demand coverage while minimizing the salary cost. An individual's *roster line* can be viewed as a sequence of *days on* and *days off*, where each day on contains a single *shift* identified by a label such as 'day', 'evening' or 'night'.

Coverage constraints imply how many nurses of appropriate skills have to be scheduled for each demand period. For ease of exposition we consider all nurses equally-skilled throughout the rest of this paper. To make the model described hereafter suitable for the more general case of the skill-dependent coverage constraints, one would have to generate the new roster lines separately for each skill type. Also the operating room scheduling procedure would have to take into account the differences in skills between the nurses, however, the structure of the problem remains unchanged and, hence, can easily be adapted.

Collective agreement requirements are rules that define acceptable schedules for individual nurses in terms of total workload, weekends off and shift transitions (e.g. a morning shift after a night shift is not allowed). These rules cannot be violated and dramatically reduce the set of feasible individual

roster lines. We make abstraction of differences in individual, nurse-specific preferences and only consider those restrictions which are stated in the collective agreement rules and consequently apply on all nurses. The more general case of heterogeneous nurses would only affect the NSP roster line pricing problem.

In what follows we state the standard set covering model, which is often used for this type of problem. Let J be the set of feasible roster lines j and I be the set of demand periods i. Let $d_i \in \Re^+ \ \forall i \in I$, denote the required number of nurses scheduled during period i. Furthermore, let a_{ij} be 1 if roster line j contains an active shift during period i and 0 otherwise. The general integer decision variable $x_j \ \forall j \in J$, indicates the number of individual nurses that are scheduled by roster line j. Then, the nurse scheduling problem (NSP) can be stated as follows:

$$\text{Minimize} \quad \sum_{j \in J} x_j \tag{3.1}$$

$$\text{Subject to:} \quad \sum_{j \in J} a_{ij} x_j \geqslant d_i \quad \forall i \in I \tag{3.2}$$

$$x_j \in \{0, 1, 2, \ldots\} \quad \forall j \in J. \tag{3.3}$$

3.3. Solution procedure for the nurse scheduling problem

The integer program (IP) (3.1)–(3.3) is solved by first solving the linear programming relaxation and then using a branching scheme to drive the solution into integrality. As the number of possible roster lines an individual can work is usually too large to allow complete a priori enumeration, column generation is often applied to solve the LP relaxation. Typically, the pricing step involves the solution of a dynamic programming shortest path problem (also called the *subproblem*) to find the legal column with the most negative reduced cost. Let $\pi_i \ \forall i \in I$, denote the dual price of constraint (3.2). Then, the reduced cost of a new column (roster line) j is given by:

$$1 - \sum_{i \in I} a_{ij} \pi_i. \tag{3.4}$$

The process of adding new columns continues until no more columns *price out*, i.e., no more columns with negative reduced cost can be found. However, at that point the solution is not necessarily integral and applying a standard branch-and-bound procedure to the restricted master with its existing columns will not guarantee an optimal (or feasible) solution. Therefore, a branching scheme has to be applied to drive the solution into integrality. After branching, new columns might price out favorably and hence have to be added to the model.

Barnhart et al. (1998) discuss appropriate branching strategies for solving a mixed integer program (MIP) using column generation. Since NSP (3.1)–(3.3) has identical restrictions on subsets (i.e., there are no subsets having a separate convexity constraint), elaborating a branching scheme is a complex issue. Conventional integer programming branching on variables is not effective for reasons of symmetry and also because fixing variables destroys the structure of the subproblem. Vanderbeck and Wolsey (1996) developed a general rule in which one is branching on the constraints (see also Vanderbeck, 2000). The drawback is that the branching constraints cannot be used to eliminate variables and have to be added to the formulation explicitly. Hence, each branching constraint will contribute an additional dual variable to the reduced cost, complicating the pricing problem. Since it does not lie in the scope of this work to discuss effective branching schemes for the NSP, we will not go into further details about this, but instead refer the reader to the specialized literature.

3.4. The generalized nurse scheduling problem

In the NSP the right-hand side values of the coverage constraints (i.e., the d_i's in formulation (3.1)–(3.3)) are considered to be fixed. Nevertheless, coverage constraints are based on workload estimations that entail the summations of individual patient *workload contributions*. An individual patient workload contribution is determined by the *patient type*. The patient type can generally be described by three dimensions. The first dimension is the type of surgery the patient has undergone. The second is the number of periods the patient has already recovered. The third is the period to which the workload applies. For instance, some ailments may require increased care during nights.

Instead of assuming the demand values to be fixed, the GNSP considers them to be dependent on the number and type of patients undergoing surgery in the hospital at each moment. By manipulating the master surgery schedule hospital management can create (and choose between) a number of different workload distributions, further referred to as *workload patterns*. Let K denote the set of possible

workload patterns that could be generated by modifying the surgery schedule. These will be obtained by enumerating all possible ways of assigning operating blocks to the different surgeons, subject to surgery demand and to capacity restrictions (for more details see Section 4.2). Each workload pattern k is described by a number of periodic demands $d_{ik} \in \{0,1,2,\ldots\}\ \forall i \in I$. Let z_k be 1 if the surgery schedule that corresponds to workload k is chosen and 0 otherwise. Then, the problem can be stated as follows:

$$\text{Minimize} \quad \sum_{j\in J} x_j \tag{3.5}$$

$$\text{Subject to:} \quad \sum_{j\in J} a_{ij}x_j \geqslant \sum_{k\in K} d_{ik}z_k \quad \forall i \in I, \tag{3.6}$$

$$\sum_{k\in K} z_k = 1, \tag{3.7}$$

$$x_j \in \{0,1,2,\ldots\} \quad \forall j \in J, \tag{3.8}$$

$$z_k \in \{0,1\} \quad \forall k \in K. \tag{3.9}$$

3.5. Solution procedure for the generalized nurse scheduling problem

In this part, we show that the column generation approach to solve the LP relaxation of NSP can easily be extended to cope with the GNSP. Similarly to the roster lines, the number of possible workload patterns is usually too large to allow for complete a priori enumeration. Also here the process starts with a limited subset of workload patterns and new patterns (columns) are added as needed. Therefore, a second subproblem has to be solved. The generation of a new workload pattern boils down to the construction of a new master surgery schedule. The subproblem is constrained by a set of specific surgery schedule restrictions. Its objective is the minimization of the reduced cost of a new workload pattern. Let γ denote the dual price of the workload pattern convexity constraint (3.7). Then the reduced cost of a new workload pattern k is given by:

$$0 - \gamma + \sum_{i\in I} \pi_i d_{ik}. \tag{3.10}$$

Obviously, the appropriate solution approach to price out a new workload pattern strongly depends on the characteristics of the master surgery schedule. In this paper, the workload pattern pricing problem is formulated as an IP and solved using a state-of-the-art optimization package (CPLEX). More details on this formulation can be found in Section 4.2.

4. Pricing problems

4.1. Generating a new roster line

Although the generation of a new roster line happens in a standard way (shortest path problem with side constraints solved with recursive dynamic programming) (see, e.g. Caprara et al., 2003) and its exact implementation is not really necessary for understanding the general idea of this paper, we briefly discuss the procedure. First, we summarize the restrictions that apply to a roster line.

We take into account five types of requirements when building a new roster line. First of all, a nurse cannot work more than one shift per day. Second, the overall number of *active days*, i.e., days in which the roster line contains an *active shift* (*day*, *evening* or *night*), cannot exceed a certain limit. Third, the maximum number of *consecutive* working days is also constrained. The same holds for the maximum number of consecutive rest days. A sequence of working days is further referred to as a *block*. Fourth, the number of the so-called unpopular shifts (night shifts, weekend shifts) is limited per roster line. Fifth, in a block certain shift transitions are not allowed. For instance, a nurse cannot switch from, say, a night shift to a morning shift without having a rest first.

Generating a new roster line is typically done using a dynamic programming recursion. To this aim, we define a table giving the minimum cost that can be achieved in days 1 to d by a roster line that, starting from a situation in which on day d a shift s is scheduled and in which between days d to n a certain number of active shifts f occurred, a certain number of unpopular shifts g occurred and a number of consecutive working or rest days h (including day d) is assigned. Formally, the entries of the table are of the form

$$\tau(d,f,g,s,h), \tag{4.1}$$

defined for $d = 1,\ldots,n$, $f = 0,\ldots,f_{\max}$, $g = 0,\ldots,g_{\max}$, $s \in S$, $h = 0,\ldots,h_{\max}$ where n denotes the number of days in the scheduling horizon, $f_{\max}$ denotes the maximum number of working days in a roster line, $g_{\max}$ is the maximum penalty in terms of unpopular shifts, S is the set of shift types (“*day*, *evening*, *night*, *rest*”) and $h_{\max}$ is the maximum of

both the maximum number of consecutive working days ($h_1^{\max}$) and the maximum number of consecutive rest days ($h_2^{\max}$).

The computation of the entries in the table is done by starting at the beginning of the time horizon and by working forward by considering an insertion of a shift type s on the next day d of the roster line associated with an entry already computed. Before starting the recursion all entries of table $\tau(d,f,g,s,h)$ are initialized to 999999999. The minimal reduced cost of a new roster line can now easily be calculated by starting the recursion on day n and minimizing over each shift type. Once all the calculations are done, the best new roster line can easily be constructed backward. The overall space complexity of the dynamic programming recursion is

$$\mathrm{O}(n \cdot f_{\max} \cdot g_{\max} \cdot |S| \cdot h_{\max}) \tag{4.2}$$

whereas the time complexity is (in the case that there are no forbidden shift transitions),

$$O(n \cdot f_{\max} \cdot g_{\max} \cdot |S|^2 \cdot h_{\max}) \tag{4.3}$$

since each entry of the table is updated by considering up to O(| S |) other entries.

4.2. Generating a new workload pattern

A new workload pattern can be obtained by building a new surgery schedule. The capacity preserved for the different surgeons (or, more generally, surgery groups) is already determined by the case mix planning and considered to be fixed in our application. Elective case scheduling is also left out of consideration for two reasons. First of all, the impact of each specific elective case on the workload is rather limited. It is the type of surgery that determines the workload contribution, not the individual case. Second, it is very hard to predict the precise impact of the individual cases on the workload contribution at the moment that the nurse rosters have to be built. Often, at that moment, an important part of the elective surgery scheduling is still to be done.

This work is concerned with *cyclic* master surgery schedules. Cyclic schedules are schedules that are repeated after a certain time period (referred to as the cycle time). During such a cycle time there might be a number of time periods during which surgery cannot take place. These periods are referred to as the inactive periods, the others are active. Typically, cycle times are multitudes of weeks in which the weekends are inactive periods.

In our application, a new surgery schedule is built by solving an integer program. To find a new workload pattern with minimal reduced cost given the current set of roster lines and workload patterns, the objective function minimizes the dual price vector of the demand constraints (3.6) multiplied by the new demands. We deal with two types of constraint. Surgery demand constraints determine how many blocks must be preserved for each surgeon. Capacity constraints ensure that the number of blocks assigned during each period do not exceed the available capacity. Let y_{rt} ($\forall r \in R$ and $t \in T$) be the number of blocks assigned to surgeon r in period t where T represents the set of active periods and R the set of surgeons. Let q_r be the number of blocks required by each surgeon r. Let b_t be the maximal number of blocks available in period t. Let $w_{rti} \in \Re^+$ denote the contribution to the workload of demand period i of assigning one block to surgeon r in period t. Then, the integer program to construct a new surgery schedule (and at the same time price out a new workload pattern k) is as follows:

$$\text{Minimize} \quad \sum_{i \in I} \pi_i d_{ik} \tag{4.4}$$

$$\text{Subject to:} \quad \sum_{t \in T} y_{rt} = q_r \quad \forall r \in R, \tag{4.5}$$

$$\sum_{r \in R} y_{rt} \leqslant b_t \quad \forall t, \in T \tag{4.6}$$

$$\sum_{r \in R} \sum_{t \in T} w_{rti} y_{rt} \leqslant d_{ik} \quad \forall i, \in I \tag{4.7}$$

$$y_{rt} \in \{0,1,2,\ldots,\min(q_r,b_t)\} \quad \forall r \in R, \ \forall t \in T, \tag{4.8}$$

$$d_{ik} \in \{0,1,2,\ldots\} \quad \forall i \in I. \tag{4.9}$$

The objective function (4.4) minimizes the reduced cost of a new workload pattern. Observe that the periodic demands d_{ik} are now an integral part of the decision process, whereas these are merely coefficients in the master problem (3.5)–(3.9). Constraint set (4.5) implies that each surgeon obtains the number of required blocks. Constraint set (4.6) ensures that the number of blocks assigned does not exceed the available number of blocks in each period. Constraint set (4.7) triggers the d_{ik}'s to the appropriate integer values. Finally, constraint sets (4.8) and (4.9) define y_{rt} and d_{ik} to be integer.

At first sight, constraint set (4.9) which requires the periodic demands d_{ik} to be integral seems to be redundant from a formulation point of view.

Indeed, due to constraint (3.6) and the fact that $a_{ij} \in \{0,1\}$ and $x_j \in \{0,1,2,\ldots\}$ fractional demand values d_{ik} would also be covered by the upper integer number of nurses. The reason why we require the d_{ik}'s to be integral is to improve the computational efficiency of the overall branch-and-price algorithm. We come back to this issue in Section 6.1.

5. Overview of the branch-and-price algorithm

Fig. 2 gives a schematic overview of the branch-and-price algorithm to solve the GNSP.

The algorithm starts with a heuristic in order to find an initial solution. The heuristic generates only one workload pattern. This is done by building a surgery schedule for which the sum of the resulting quadratic demand values is minimized. The idea is to level the workload distribution as much as possible over the time horizon and as such to avoid the occurrence of peaks in the workload. This approach turned out to be beneficial for the surgery scheduling problem in which the expected shortage of beds has to be minimized (see Beliën and Demeulemeester, 2006). The surgery schedule is built with a mixed integer program (MIP) in which the constraints are given by (4.5)–(4.8) (replacing the d_{ik}'s by d_i's) and the objective is:

$$\text{Minimize} \quad \sum_{i \in I} d_i^2 \tag{5.1}$$

with d_i the required number of nurses in period i. To speed up the heuristic, the d_i's are not required to be integral. Instead, we round each d_i to the next upper integer after solution of the quadratic MIP. Given this workload pattern, new roster lines are added until the set of roster lines (one nurse scheduled by each roster line) completely satisfies the coverage constraints. A new roster line is found by solving exactly the same shortest path problem as in Section 4.1, but replacing the dual prices π_i by the remaining right-hand side values d_i. As such each new roster line cuts the peaks in the remaining workload pattern until all demand is covered.

After detection of an initial solution, the objective value is saved as an upper bound and both the surgery schedule and the nurse schedule are registered. The columns making up the initial solution are entered into the master together with a supercolumn for each coverage constraint, i.e., a variable having a one for the corresponding constraint in the constraint matrix, but also a very large cost in the objective function. Hence, the supercolumns will never be chosen in a final solution, however, can always be used to find a feasible solution of the master in each stage of the branch-and-bound algorithm.

The algorithm starts with the LP optimization loop in which, iteratively, a number of new roster lines and one new workload pattern are added until no more columns price out. Observe that the roster lines are added until no more lines with negative reduced cost can be found, whereas only one workload pattern is generated, after which the generation of new roster lines restarts. This approach turned out to be the most successful, given the generally larger computation times to price out a new workload pattern.

Upon detection of the LP optimum, the solution is checked for fractional z_k's (workload patterns). If

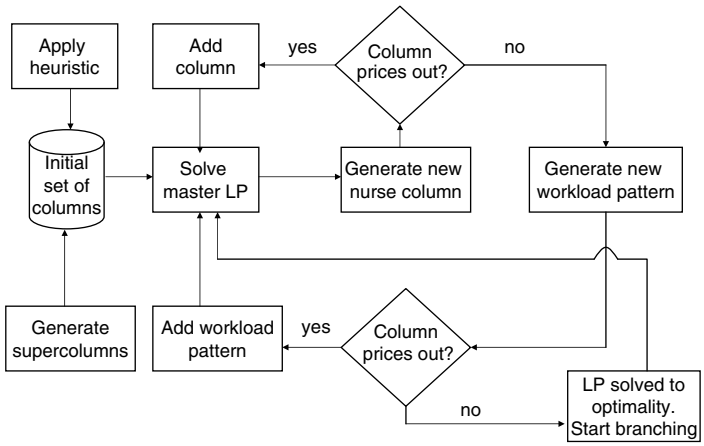

Fig. 2. Schematic overview of the GNSP branch-and-price algorithm.

there still are fractional z_k's, branching is applied in order to drive the solution into an integral z solution (i.e., with only one z_k equal to 1 and all other equal to 0). The algorithm does not branch until an integral x_j (roster line) solution, because branching schemes for the x_j variables are not straightforward to implement and significantly complicate the roster line subproblem. Moreover, it provides no extra value for the extended model, which is the subject of this paper. Instead, we report lower and upper bounds for the required number of nurses to cover demand. The lower bound is the best possible solution with exactly one z_k equal to 1, however one for which the x_j's are not necessarily integral. Hence, the solution represented by the lower bound might not be interpretable in terms of the nurse schedule (e.g., schedule 2.5 nurses following roster line j). The upper bound on the other hand is the best found overall integer solution (with also integrality of the x_j's), which is fully interpretable.

In order to increase the lower bound as much as possible, the branch-and-bound tree is traversed in a best-search way. After each move in the tree, the master problem is solved with required integrality on both the x_j's and the z_k's. Because the integral master problem is often computationally very intensive, the MIP optimizer is interrupted after a specified time interval (e.g., 10 s). If a better solution is found, the upper bound decreases and as such the gap between the lower and upper bound tightens.

This work is only concerned with a branching scheme for driving the z_k's to integrality and leaves the x_j's out of consideration. We apply a constraint branching scheme (Ryan and Foster, 1981) which works as follows.

First we search for the highest fractional z_k. Let this be $z_{k'}$. Then we select another $z_k > 0$, say $z_{k'}$, and take the first period i for which $d_{ik'} \neq d_{ik''}$. If no such period exists, both z_k's represent essentially the same workload patterns and hence one of them can be set to 0 while its fractional value is added to the other one. Suppose we found period i' as the branching period with $d_{i'k'} < d_{i'k''}$. Then, we create two nodes in the branch-and-bound tree. In the left node, we imply $d_{i'k} \leqslant d_{i'k'}$ and in the right node we imply $d_{i'k} \geqslant d_{i'k'} + 1$. Fig. 3 visualizes this branching scheme. Else if $d_{i'k'} > d_{i'k''}$ we imply $d_{i'k} \leqslant d_{i'k''}$ in the left node and $d_{i'k} \geqslant d_{i'k''} + 1$ in the right node.

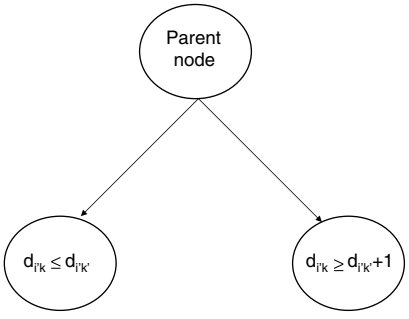

Fig. 3. Binary branching scheme in the case of $d_{i'k'} < d_{i'k''}$.

6. Computational performance issues

In this section, we present some techniques that helped to improve the computational efficiency of the algorithm.

6.1. Integral versus fractional demand values

It has already been mentioned at the end of Section 4.2 that we imply the d_{ik}'s to be integral in the workload pattern pricing problem. Although this is not necessary from a formulation point of view, it has a substantially positive impact on the overall computational efficiency of the algorithm.

Implying integrality of the d_{ik}'s affects the computation time in two ways. On the one hand, there is a negative impact, because the pricing problem itself becomes more complex. On the other hand, there is a positive impact as far fewer columns can be found with negative reduced cost. Preliminary results indicate that this positive effect dramatically exceeds the negative effect. Consequently, the master LP is solved much faster when integrality of the d_{ik}'s is implied. Moreover, requiring integral demand values in the workload patterns makes the LP optimal solution substantially less fractional in terms of the x_j's. Hence, finding a global optimum (with both integrality on the z_k's and on the x_j's) turns out to be much easier. In our application the gap between the lower and upper bound becomes much smaller.

6.2. Upper bound pruning for the workload pattern pricing problem

Basically, we are no longer interested in finding the column with the lowest reduced cost from the moment we know that this reduced cost will be positive anyway. Hence, we can act as if we already found a solution with reduced cost 0 by providing an appropriate upper bound. For the workload

pattern subproblem, this observation yields dramatic time savings.

The reduced cost expression (3.4) consists of a fixed part and a variable part. By setting the upper bound equal to the fixed part with reverse sign, we act as if we found already a new column with reduced cost equal to 0. The reduced cost of a workload pattern is given by $0-\gamma+\sum_{i\in I}\pi_i d_{ik}$. Consequently, we provide γ as an upper bound in the integer program (4.4)–(4.9).

Note that, since generating a new roster line is done with a dynamic programming approach using backward recursion, upper bound pruning cannot be applied here. As an alternative, we wrote an enumeration algorithm including both dynamic pruning and pruning based on bound comparisons. Dynamic pruning occurs if a state has already been visited at lower cost. For pruning based on bound comparisons we need an upper and a lower bound for the best new roster line. Since the reduced cost of a new roster line is given by $1-\sum_{i\in I}a_{ij}\pi_i$, we can provide -1 as an initial upper bound. Obviously, this bound is decreased each time a better roster line is found. Starting from a certain day, a lower bound on the minimal cost path could be obtained by selecting for each remaining day the shift with the lowest total of corresponding dual prices, i.e.,

$$\mathrm{MIN}\left\{\min_{s\in S\setminus\{\mathrm{rest}\}}\{\lambda_{d,i}\},0\right\}\quad\forall d \tag{6.1}$$

and summing up only the $(f_{\max}-f)$ lowest values amongst these. In other words, for calculating the lower bound, we relax all constraints but the not-more-than-one-shift-per-day constraint and the maximum number of active days constraint. Preliminary tests, however, indicated that this enumeration approach is outperformed by the dynamic programming approach.

6.3. Two-phase approach for the workload pattern pricing problem

During the LP optimization loop it is not necessary to find the column with the most negative reduced cost, any column with negative reduced cost will do. Again, particularly for the computationally intensive workload pattern pricing problem, using this observation dramatically decreases the computation times. To guarantee optimality of the LP solution, a two-phase approach is applied for the workload pattern pricing problem. In the first phase, a certain time limit is set for the MIP optimizer. Only if no new workload pattern is found with negative reduced cost within this time limit, the algorithm enters the second phase. In this phase the time limit is undone and the optimizer is required to search until a feasible solution is found with negative reduced cost or it is proven that such a column does not exist.

6.4. Lagrange dual pruning

It is well known that Lagrangian relaxation can complement column generation in that it can be used in every iteration of the column generation scheme to compute a lower bound to the original problem with little additional computational effort (see, e.g., Van den Akker et al., 2002; Vanderbeck and Wolsey, 1996). If this lower bound exceeds an already found upper bound, the column generation phase can end without any risk of missing the optimum. Using the information from solving the reduced master and the information provided by solving the pricing problem for a new workload pattern k, it can be shown (see, e.g., Hans, 2001) that a lower bound is given by $\delta+RC_k\theta_k$ where δ is the objective value of the reduced master, RC_k is the reduced cost of a newly found workload pattern k and θ_k is a binary variable equal to 1 when RC_k is negative and set to zero, otherwise. This lower bound is referred to as the Lagrangian lower bound, since it can be shown that it equals the bound obtained by Lagrange relaxation.

Obviously, if the pricing procedure finds a negative reduced cost column during the first phase and hence does not enter the second phase (see Section 6.3) this lower bound cannot be used, because the workload pattern pricing problem has not been solved to optimality.

Using CPLEX, it is very easy to set upper bounds, time limits and limits on the number of feasible solutions. Moreover, it can easily be verified if either the problem has been solved to optimality or optimization has prematurely ended because of an insufficient time limit.

7. Results

7.1. Test set

To test the algorithm, we started from the same set as the one introduced in Beliën and Demeulemeester (2006) for testing the surgery scheduling level-

Table 1
Factor settings in surgery scheduling test set

Factor setting	Number of blocks per day	Number of surgeons	Division of requested blocks	Number of patients per surgeon	LOS
1	3–6	3–7	Evenly distributed	3–5	2–5
2	7–12	8–15	Not evenly distributed	3–12	2–12

ing algorithms. All surgery scheduling problems in this set involve a cycle time of 7 days. The last two days are not available to allocate operating room time (weekend), which is common practice. The problems differ with respect to five factors. These are as follows: (1) the number of time blocks per day, (2) the number of surgeons, (3) the division of requested blocks per surgeon, (4) the number of operated patients per surgeon and finally (5) the length of stay (LOS) distribution. If we consider two settings for each factor and repeat each factor combination three times, we obtain $2^5 * 3 = 96$ test instances. Table 1 contains the settings for these five factors. Some of the factor settings require some further explanation.

The number of blocks per day is drawn from a uniform distribution with bounds 3 and 6 in the first setting and 7 and 12 in the second setting. A block is defined as the smallest time unit for which a specific operating room can be allocated to a specific surgeon (or surgical group). Note that, due to large set-up time and costs, in real-life applications the number of blocks per day in one operating room is usually 1 or 2, i.e. each surgical group has the operating room for at least half a day. Hence, considering more blocks can be seen as a way of considering more operating rooms as there is no difference from a computational point of view. The third factor indicates whether or not the requested blocks are evenly distributed among all surgeons; e.g., if there are 20 time blocks and five surgeons, each surgeon requires four time blocks in the evenly distributed case, whereas in the unevenly distributed case huge differences can occur. For the LOS in factor 5 we used discrete multinomial distributions with mean dependent on the factor setting and having the shape of a continuous exponential distribution. This test set is available from the author's website (http://www.econ.kuleuven.be/public/NDBAE13/GNSP.zip).

Next, we generated some weights w_{rti} defining the contributions to the workload of period i of allocating a block to surgeon r in period t. These weights vary linearly with the number of patients of surgeon r operated in period t that are still in the hospital in period i. The patient's workload contribution generally decreases the longer the patient has already recovered in the hospital. In our test set the workload demand periods coincide with the shifts. Furthermore, we set the contribution to a 'day' shift two times as large as the one to an 'evening' shift and four times as large as the one to a 'night' shift. Obviously, although attempting to represent realistic scenarios, these contributions are chosen somewhat arbitrarily.

Third, we composed a set of collective agreement rules which apply on individual roster lines. The scheduling horizon amounted to 4 weeks or 28 days ($= n$). The maximum number of days an active shift could be scheduled ('day', 'evening' or 'night') was set to 20 ($= f_{\max}$). Shifts during the weekends were marked as unpopular shifts: day and evening shifts got a penalty of 1, night shifts got a penalty of 2. The maximum number of consecutive working days was set to 6 ($= h_1^{\max} = h_{\max}$) and the maximum number of consecutive rest days was set to 3 ($= h_2^{\max}$). Furthermore, we distinguished between two scenarios: a hard constrained scenario and a flexible one. Collective agreement rules in the hard constrained scenario differ from those in the flexible scenario on the following two points:

- In the hard constrained scenario there is only one shift type allowed within each block. In other words, no shift transitions between different shift types can occur without scheduling a rest first. In the flexible scenario all shift transitions are allowed, except the following three: a 'night' shift followed by a 'day' shift, a 'night' shift followed by an 'evening' shift or an 'evening' shift followed by a 'day' shift.
- In the hard constrained scenario the maximal penalty with respect to unpopular shifts is set to 4, whereas in the flexible scenario it is set to 8 ($= g_{\max}$).

The branch-and-price algorithm was coded in C++ and linked with the CPLEX callable optimization library version 8.1 (ILOG, 2002). The tests were

Table 2
Lower and upper bounds for the NSP and the GNSP

No.	Problem	Flexible scenario				Hard constrained scenario			
		NSP		GNSP		NSP		GNSP	
		lb	ub	lb	ub	lb	ub	lb	ub
1	d00000_0	15	17	13	15	19	19	16	17
2	d00000_1	26	28	25	27	34	35	31	31
3	d00000_2	25	27	23	25	32	32	28	29
4	d00001_0	40	42	39	41	49	50	47	48
5	d00001_1	45	47	44	46	54	54	52	53
6	d00001_2	94	96	92	94	112	113	109	110
7	d00010_0	34	36	32	35	43	43	40	40
8	d00010_1	40	42	38	40	49	50	47	47
9	d00010_2	28	30	26	27	34	35	32	33
...	...	...	...	...	...	...	...	...	...
88	d11101_0	96	98	94	96	114	115	112	113
89	d11101_1	99	102	97	99	119	120	116	116
90	d11101_2	122	125	119	121	145	146	142	143
91	d11110_0	83	85	80	82	101	102	96	96
92	d11110_1	111	113	109	111	138	139	132	132
93	d11110_2	58	60	56	58	73	74	67	68
94	d11111_0	252	254	249	252	303	304	296	297
95	d11111_1	119	122	116	119	143	144	139	140
96	d11111_2	135	137	131	133	162	163	156	157
	Average	70.18	72.43	68.33	70.44	86.07	86.73	81.91	82.61

done on a 2.4 GHz Pentium 4 PC under the Windows XP operating system.

7.2. Savings

Table 2 contains the lower and upper bounds for both the NSP and the GNSP. In the NSP, a surgery schedule is generated randomly. The resulting workload pattern contains the (fixed) right-hand side values of the coverage constraints. Then, the NSP is solved using column generation. In the GNSP new surgery schedules (and hence resulting workload patterns) are generated during search if needed. We distinguish between the flexible and the hard constrained scenario. To give an idea of the variability, the detailed bounds are provided for the first 9 and the last 9 problems of the problem set. The last line contains the average bounds over the whole set. Observe that the name of each problem (d*ijklm_n*) contains the information about the surgery scheduling subproblem: *i* stands for the setting of the first factor in Table 1 (0 for the first setting, 1 for the second), *j* for the second one, etc..., and *n* for the iteration number.

From these results, one may conclude the following. First have a look at the upper bounds, which are after all the solutions that will be worked with. Although it is not guaranteed that the upper bound will be better (one might be lucky in the NSP and find the same or even a better overall integer solution), the upper bounds for the GNSP are generally better than those for the NSP. We compared them using a one-tailed paired *T*-test. The extremely small *p*-values obtained indicate that the differences are statistically significant both for the flexible and for the hard constrained case. The same results are obtained for the lower bounds. Unlike the upper bounds, the GNSP lower bounds are of course guaranteed to be at least as good as the NSP lower bounds.

When comparing the lower bounds for the NSP with the upper bounds for the GNSP, both scenarios entail different conclusions. The average lower bound for the NSP is lower than the average upper bound for the GNSP in the flexible scenario, whereas the reverse is true in the hard constrained scenario. Both differences turned out to be significant using a one-tailed paired *T*-test (again extremely small *p*-values). This observation can easily be explained. The stricter the collective agreement rules, the harder it is to nicely fit the nurse rosters into the required workload pattern in the NSP. As the workload pattern can be adapted in the GNSP, the GNSP includes more possible savings in the case of severe collective agreement requirements.

7.3. Interpretation of the savings

In the previous section, we concluded that integrating the surgery scheduling process with the nurse scheduling process may yield important savings in terms of the required nurses to hire. It is relatively easy to identify the shifts during which too many nurses are working. However, identifying and quantifying the main cause is much more difficult. In this paragraph, it is shown how the results from the integrated approach can be used to quantify the shares of the two types of waste.

First, an unfavorable workload pattern may contain many workload demands that slightly exceed the workforce of x nurses, but that are dramatically inferior to the workforce of $x+1$ nurses. In terms of the d_{ik}'s one could think of many d_{ik}'s having a small decimal part, e.g., 6.1, 8.2, 4.05, etc... This type of waste is referred to as the waste due to the workforce surplus per shift. In many hospitals this kind of waste is taken care of by simply scheduling x nurses instead of $x+1$ nurses during those shifts. The result is a group of overworked nurses and an almost certain decrease in the quality of care. This illustrates how the GNSP approach can also be very useful for optimizing qualitative instead of quantitative objectives.

Second, waste also originates from the inflexibility of the roster lines, due to strict general agreement requirements. Because of this, no set of roster lines can be found that perfectly fit with the workload demand. This source of waste is further referred to as waste due to the inflexibility of roster lines.

Table 3 gives an overview of the importance of both sources of waste. We again distinguish between the flexible scenario and the hard constrained scenario. For each scenario there are three columns. The first column contains the total waste in terms of overstaffing in the NSP compared with the GNSP. These numbers are obtained by subtracting the upper bounds for the GNSP from those for the NSP. The second and third columns indicate the parts of this total waste that are due to the workforce surplus per shift and to the inflexibility of roster lines. These numbers can easily be calculated as follows. First, for both the NSP and the GNSP we make the sum of the (integral) demands of the chosen workload pattern. Call this number the total required workforce ($=\sum_{i\in I} d_i$ for the NSP and $\sum_{i\in I}\sum_{k\in K} d_{ik}z_k$ for the GNSP). Next, divide this number by the workforce per nurse ($=f_{\max}$ in our application). This gives the minimal number of nurses that would be needed and can be obtained in the case of fully flexible roster lines.

Table 3
Interpretation of the savings

No.	Problem	Total waste	Flexible scenario		Total waste	Hard constrained scenario	
			Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines		Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines
1	d00000_0	2	1.2	0.8	2	1.2	0.8
2	d00000_1	1	1.2	−0.2	4	1.4	2.6
3	d00000_2	1	2	−1	3	1	2
4	d00001_0	1	1.2	−0.2	2	0.6	1.4
5	d00001_1	2	1	1	1	0.2	0.8
6	d00001_2	2	1.6	0.4	3	0	3
7	d00010_0	1	1.4	−0.4	3	1	2
8	d00010_1	1	1.6	−0.6	3	1.6	1.4
9	d00010_2	1	1.8	−0.8	2	−0.6	2.6
...	...	...	...	...	...	...	...
88	d11101_0	2	1.4	0.6	2	0.6	1.4
89	d11101_1	2	1.8	0.2	4	0.2	3.8
90	d11101_2	1	2.2	−1.2	3	0.2	2.8
91	d11110_0	2	1.6	0.4	6	0.8	5.2
92	d11110_1	2	0.8	1.2	7	0.6	6.4
93	d11110_2	1	2	−1	6	1.8	4.2
94	d11111_0	2	1.2	0.8	7	0.2	6.8
95	d11111_1	2	1.8	0.2	4	−0.6	4.6
96	d11111_2	1	2	−1	6	0.6	5.4
	Average	1.58	1.43	0.16	4.11	0.28	3.84

The difference between these numbers for the NSP and GNSP is the waste due to the workforce surplus per shift. The difference between the total waste and the waste due to the workforce surplus per shift is the waste due to the inflexibility of roster lines. Observe that these wastes may be negative (e.g., the waste due to workforce surplus per shift for problem d00000_2 is −1). This situation occurs when the gain with respect to one source of waste is so large that the best found solution for the GNSP includes a limited sacrifice with respect to the other source of waste.

The results in Table 3 clearly indicate that the importance of the source of waste strongly depends on the strictness of the general agreement requirements. The stricter these requirements are, the larger is the share of the waste due to the inflexibility of the roster lines.

7.4. Computational results

Tables 4 and 5 contain the computational results for the flexible respectively hard constrained scenario. For the NSP, both the computation time and the number of generated roster lines are given. For the GNSP also the number of generated demand patterns and the number of nodes in the branch-and-bound tree are provided.

Obviously, the required computation times for the GNSP exceed those for the NSP. However, taking into account the explosion of the feasible solution space for the GNSP compared to the NSP, the increase in computation time is rather small. We can conclude that column generation is a viable technique for solving the GNSP.

If we compare the flexible scenario with the hard constrained scenario, a couple of things attract our attention. First of all, observe that for the NSP the computation times for the flexible scenario surpass those for the hard constrained scenario, whereas for the GNSP the computation times for the hard constrained scenario exceed those for the flexible scenario. For the NSP this difference is statistically significant (extremely small p-value for a two-tailed paired T-test) and easy to explain. In the flexible scenario, much more legal roster lines exist and hence much more roster lines with negative reduced cost are found during the search process (on average 207.25 versus 106.07). Moreover, the time needed to price out a new roster line is also larger since the feasible state space contains more legal states.

For the GNSP the difference in computation time is not statistically significant at the 5% level (p-value of 0.113 for a two-tailed paired T-test). As again the number of generated roster lines is significantly smaller (very small p-value for a two-tailed paired

Table 4
Computational results for the flexible scenario

Nr.	Problem	NSP		GNSP			
		Time (milliseconds)	Roster lines	Time (milliseconds)	Roster lines	Workload patterns	Nodes
1	d00000_0	43484	150	44,422	183	2	0
2	d00000_1	44063	174	51,000	196	2	0
3	d00000_2	46423	235	45,438	213	2	0
4	d00001_0	44078	173	46,000	221	2	0
5	d00001_1	43829	167	45,172	190	2	0
6	d00001_2	44844	212	48,829	238	3	0
7	d00010_0	45266	211	70,359	274	2	0
8	d00010_1	46311	237	185,623	535	17	8
9	d00010_2	44594	208	166,892	640	32	13
...	...	...	...	...	...	...	...
88	d11101_0	44390	213	47,984	243	2	0
89	d11101_1	44953	228	52,031	257	2	0
90	d11101_2	44734	230	56,438	280	2	0
91	d11110_0	46203	252	358,811	555	30	15
92	d11110_1	45265	238	1,765,257	815	128	59
93	d11110_2	47359	200	423,125	507	28	14
94	d11111_0	46360	347	69,266	381	2	0
95	d11111_1	45719	243	59,063	319	2	0
96	d11111_2	45048	237	251,970	512	14	6
	Average	44146.04	207.25	99008.57	310.31	5.93	1.95

Table 5
Computational results for the hard constrained scenario

No.	Problem	NSP		GNSP			
		Time (milliseconds)	Roster lines	Time (milliseconds)	Roster lines	Workload patterns	Nodes
1	d00000_0	453	46	66,953	263	8	4
2	d00000_1	500	70	55,359	304	18	6
3	d00000_2	422	64	11,781	111	2	0
4	d00001_0	468	77	609	81	2	0
5	d00001_1	453	74	687	95	3	0
6	d00001_2	672	120	782	127	2	0
7	d00010_0	4250	113	216,064	470	79	43
8	d00010_1	953	113	323,236	448	129	47
9	d00010_2	750	80	201,970	459	102	39
...	...	...	...	...	...	...	...
88	d11101_0	2125	122	1656	130	2	0
89	d11101_1	1531	126	2625	146	2	0
90	d11101_2	1610	149	2109	159	2	0
91	d11110_0	1938	123	456,191	439	58	17
92	d11110_1	1500	152	1,228,851	508	92	45
93	d11110_2	5438	101	102,470	310	10	1
94	d11111_0	8000	251	12,265	264	2	0
95	d11111_1	4859	143	19,359	185	2	0
96	d11111_2	4922	153	1,809,557	600	221	83
	Average	1215.52	106.07	153927.85	226.05	28.08	10.81

T-test), the higher computation times for the constrained scenario must be produced by the higher number of generated workload patterns and the higher number of nodes in the branch-and-bound tree. The differences in number of generated workload patterns and in nodes in the branch-and-bound tree are found to be significant (very small *p*-values for two-tailed paired *T*-tests). This can easily be explained as follows. In the flexible scenario, it is unlikely that an extra workload pattern improves the overall solution. Thanks to the flexibility in the roster lines, an already very good solution can be found using a limited set of workload patterns. In the hard constrained case on the other hand, the inflexibility of the roster lines might obstruct the detection of a good solution. In this case, it is far more likely that adding a new workload pattern improves the overall solution.

The results above suggest that the GNSP is easier to solve if the collective agreement requirements are less strict, whereas the reverse is true for the NSP. However, it must be clear that this only holds for a specific range of strictness. First of all, observe that, if there are no collective agreement requirements at all, both the NSP and the GNSP will be straightforward to solve. However, this extreme case is not realistic. Less strict transition constraints than the ones in the flexible scenario do not occur in real life, so it makes no sense to further relax these constraints. Obviously, a nurse cannot be required to work every shift in a given time horizon (e.g., a month). So, a maximum limit on the total number of active shifts (here, $f_{\max} = 20$) must also be taken into account. In the same reasoning, the maximal number of consecutive working shifts (=6) cannot be relaxed either. Given this minimal strictness on the collective agreement requirements, the required computation time to price out a new roster line increases (due to the larger search space), if we relax any of the other constraints.

The question is, however, whether this time increase can be compensated by the fact that fewer columns have to be generated. Our results indicate that this is not the case for the NSP. For the GNSP, however, the total computation time decreases provided that the other constraints are not too much relaxed. Specifically, further relaxation of the maximum unpopular shifts penalty decreases the computation time (as shown in the computational results of the paper). However, if we combine this relaxation with more relaxations (e.g., dropping the constraint on the maximal number of consecutive rest days), the further increase in the computation time needed to price out one column cannot be compensated anymore.

In conclusion, starting from the realistic assumption of a minimum of strictness on the collective agreement requirements, the NSP is always easier

to solve the more hard constraints are added (smaller solution space). For the GNSP, however, this is not always the case.

As a final remark we note that a large part of the computation time goes to the calculation of an overall feasible solution in order to detect an upper bound after each move in the branch-and-bound tree in the GNSP and at the end of the column generation process in the NSP.

8. Conclusions and further research

This paper has presented an integrated approach for building nurse and surgery schedules. It has been shown how the column generation technique, often employed for solving nurse scheduling problems, can easily be extended to cope with this integrated approach. The approach involves the solution of two types of pricing problems, the first one is solved with a standard dynamic programming approach using recursion, the second one by means of a state-of-the-art mixed integer programming optimizer. A constraint branching scheme was proposed to drive the solution into integrality with respect to the workload patterns while the integrality of the roster lines was left out of the scope of this paper. Finally, some techniques were presented that helped to improve the computational efficiency of the branch-and-price algorithm.

Our computational results indicate that considerable savings could be achieved by using this approach to build nurse and surgery schedules. We simulated a large range of surgery scheduling instances and distinguished between a flexible and a hard constrained scenario with respect to the collective agreement requirements. Our conclusions can be summarized as follows. First of all, column generation is a good technique to deal with the extra problem dimension of modifying surgery schedules. Second, the results from the integrated approach can be used to quantify the shares of the two sources of waste: waste due to the workforce surplus per shift and waste due to the inflexibility of roster lines. Third, unlike the NSP, the GNSP turns out to become harder to solve when the collective agreement requirements are more strict.

Obviously, in real-life hospital environments it is not so easy to modify the master surgery schedule. As the surgery schedule can be considered to be the main engine of the hospital, it not only has an impact on the workload distribution for nurses, but also on several other resources throughout the hospital. Think for instance about anaesthetists, equipment, radiology, laboratory tests and consultation. This observation yields a negative as well as a positive note for the reasoning in this paper. The negative note is that the possible savings obtained through integrating the nurse and the surgery scheduling process are in real-life probably much smaller, due to the smaller flexibility with which surgery schedules can be modified. The positive note is that not only savings in nurse staffing costs are possible, but also in other related resource types, by integrating the scheduling of these resources with the surgery scheduling process. This work clearly shows the benefits of integrating scheduling processes in health care environments and moreover proposes a methodology for implementing the heart of a supporting ICT infrastructure.

Possible topics for further research include the application of this approach in a real-world environment involving a detailed report on the experienced merits and pitfalls. Furthermore, the model could be further refined, for instance, by taking into account individual constraints on the nurse rosters. This might also improve the efficiency of the approach. Whenever surgery schedules are generated, they form a set of constraints to be satisfied for nurses, providing less number of shift patterns (roster lines) for each nurse, narrowing their domain (hence, the search space). Finally, it would be interesting to develop similar techniques for one or more of the other resource types stated above.

Acknowledgements

We thank the anonymous referees for their useful comments that have undoubtedly improved the quality of this paper. We acknowledge the support given to this project by the Fonds voor Wetenschappelijk Onderzoek (FWO) – Vlaanderen, Belgium Under Contract Number G.0463.04.

References

Abdennadher, S., Schlenker, H., 1999. INTERDIP – an interactive constraint based nurse scheduler. In: Proceedings of the First International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming (PACLP99), London.

Alfares, H., Bailey, J., 1997. Integrated project task and manpower scheduling. IIE Transactions 29, 711–718.

Alfares, H., Bailey, J., Lin, W., 1999. Integrating project operations and personnel scheduling with multiple labor classes. Production Planning and Control 10, 570–578.

Azaiez, M.N., Al Sharif, S.S., 2005. A 0–1 goal programming model for nurse scheduling. Computers and Operations Research 32, 491–507.

Bard, J.F., Purnomo, H.W., 2005a. A column generation-based approach to solve the preference scheduling problem for nurses with downgrading. Socio-Economic Planning Sciences 39, 193–213.

Bard, J.F., Purnomo, H.W., 2005b. Preference scheduling for nurses using column generation. European Journal of Operational Research 164, 510–534.

Bard, J.F., Binici, C., deSilva, A.H., 2003. Staff scheduling at the United States Postal Service. Computers and Operations Research 30, 745–771.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, P.H., Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329.

Beaumont, N., 1997. Scheduling staff using mixed integer programming. European Journal of Operational Research 98, 473–484.

Beliën, J., Demeulemeester, E., 2006. Building cyclic master surgery schedules with leveled resulting bed occupancy. European Journal of Operational Research 176 (2), 1185–1204.

Blake, J.T., Carter, M.W., 2002. A goal programming approach to strategic resource allocation in acute care hospitals. European Journal of Operational Research 140, 541–561.

Blake, J.T., Carter, M.W., 2003. Physician and hospital funding options in a public system with decreasing resources. Socio-Economic Planning Sciences 37, 45–68.

Blake, J.T., Donald, J., 2002. Mount Sinai hospital uses integer programming to allocate operating room time. Interfaces 32, 63–73.

Blake, J.T., Dexter, F., Donald, J., 2002. Operating room manager's use of integer programming for assigning block time to surgical groups: A case study. Anesthesia and Analgesia 94, 143–148.

Burke, E.K., De Causmaecker, P., Vanden Berghe, G., 1999. A hybrid tabu search algorithm for the nurse rostering problem. In: McKay, B., Yao, X., Newton, C.S., Kim, J., Furuhashi, T. (Eds.), Simulated Evolution and Learning, Selected Papers from the 2nd Asia-Pacific Conference on Simulated Evolution and Learning, SEAL 98, Canberra, Australia, November 1998, vol. 1585, Springer Lecture Notes in Artificial Intelligence, pp. 187–194.

Burke, E.K., Cowling, P.I., De Causmaecker, P., Vanden Berghe, G., 2001. A memetic approach to the nurse rostering problem. Applied Intelligence 15, 199–214.

Burke, E.K., De Causmaecker, P., Vanden Berghe, G., 2004. Chapter 44: Novel meta-heuristic approaches to nurse rostering problems in Belgian hospitals. In: Leung, J. (Ed.), The Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, pp. 44.1–44.18.

Burke, E.K., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H., 2004. The state of the art of nurse rostering. The Journal of Scheduling 7, 441–499.

Burke, E.K., De Causmaecker, P., Petrovic, S., Vanden Berghe, G., 2006. Metaheuristics for handling time interval coverage constraints in nurse scheduling. Applied Artificial Intelligence 9, 743–766.

Caprara, A., Monaci, M., Toth, P., 2003. Models and algorithms for a staff scheduling problem. Mathematical Programming 98, 445–476.

Cheang, B., Li, H., Lim, A., Rodrigues, B., 2003. Nurse rostering problems – a bibliographic survey. European Journal of Operational Research 151, 447–460.

Cheng, B., Lee, J., Wu, J., 1997. A nurse rostering system using constraint programming and redundant modeling. IEEE Transactions on Information Technology in Biomedicine 1 (1), 44–54.

Darmoni, S.J., Fajner, A., Mahe, N., Leforestier, A., Vondracek, M., Baldenweck, M., 1995. Horoplan: Computer-assisted nurse scheduling using constraint-based programming. Journal of the Society for Health Systems 5 (1), 41–54.

De Causmaecker, P., Vanden Berghe, G., 2003. Relaxation of coverage constraints in hospital personnel rostering. Practice and Theory of Automated Timetabling, Proceedings of the Fourth International Conference, Gent, vol. 2740. Springer, pp. 129–147.

Dowsland, K., 1998. Nurse scheduling with tabu search and strategic oscillation. European Journal of Operational Research 106, 393–407.

Hans, E.W., 2001. Resource Loading by Branch-and-price Techniques, Ph.D. Dissertation, Twente University Press, Enschede, The Netherlands.

Hans, E.W., Wullink, G., van Houdenhoven, M., Kazemier, G., 2005. Robust Surgery Loading, Technical Report Beta-wp141, Department of Operational Methods for Production and Logistics, University of Twente.

Hughes, W.L., Soliman, S.Y., 1985. Short-term case mix management with linear programming. Hospital and Health Services Administration 30, 52–60.

ILOG, 2002. ILOG CPLEX 8.1 User's Manual.

Jaumard, B., Semet, F., Vovor, T., 1998. A generalized linear programming model for nurse scheduling. European Journal of Operational Research 107, 1–18.

Jun, J.B., Jacobson, S.H., Swisher, J.R., 1999. Applications of discrete event simulation in health care clinics: A survey. Journal of the Operational Research Society 50, 109–123.

Litvak, E., Long, M.C., 2000. Cost and quality under managed care: Irreconcilable differences? The American Journal of Managed Care 6, 305–312.

Mason, A.J., Smith, M.C., 1998. A nested column generator for solving rostering problems with integer programming. International Conference on Optimisation: Techniques and Applications, 827–834.

Mehrotra, A., Murphy, K.E., Trick, M.A., 2000. Optimal shift scheduling: A branch-and-price approach. Naval Research Logistics 47, 185–200.

Meyer auf'm Hofe, H., 2001. Solving rostering tasks as constraint optimizationSelected Papers from the Third International Conference on Practice and Theory of Automated Timetabling (PATAT-2000) Konstanz, vol. 2079. Springer-Verlag, pp. 191–212.

Miller, H.E., Pierskalla, W.P., Rath, G.J., 1976. Nurse scheduling using mathematical programming. Operations Research 24, 857–870.

Mullinax, C., Lawley, M., 2002. Assigning patients to nurses in neonatal intensive care. Journal of the Operational Research Society 53, 25–35.

Musliu, N., Gärtner, J., Slany, W., 2000. Efficient generation of rotating workforce schedules. In: Proceedings of the Third International Conference on the practice and theory of automated timetabling (PATAT 2000), Konstanz, pp. 314–332.

Okada, M., Okada, M., 1988. Prolog-based system for nursing staff scheduling implemented on a personal computer. Computers and Biomedical Research 21 (1), 53–63.

Ozcan, E., 2005. Memetic algorithms for nurse rostering. In: Lecture Notes in Computer Science, Proceedings of the 20th International Symposium on Computer and Information Sciences. Spinger-Verlag, pp. 482–492.

Punnakitikashem, P., Rosenberger, J.M., Behan, D.B., 2005. Stochastic Programming for Nurse Assignment, Technical Report COSMOS 05-01, University of Texas at Arlington, Arlington, TX.

Rifai, A.K., Pecenka, J.O., 1989. An application of goal programming in healthcare planning. International Journal of Production Management 10, 28–37.

Robbins, W.A., Tuntiwongbiboon, N., 1989. Linear programming is a useful tool in case-mix management. Healthcare Financial Management 43, 114–116.

Ryan, D.M., Foster, B.A., 1981. An integer programming approach to scheduling. In: Weren, A. (Ed.), Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling. North-Holland, Amsterdam, pp. 269–280.

USDHHS, 2002. Projected Supply, Demand and Shortages of Registered Nurses: 2000–2020, National Center for Health Workforce Analysis. US Department of health and Human Services, Rockville, MD.

Van den Akker, M., Hoogeveen, H., van de Velde, S.L., 2002. Combining column generation and lagrangian relaxation to solve a single-machine common due date problem. INFORMS Journal on Computing 14, 37–51.

Vanderbeck, F., 2000. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research 48, 111–128.

Vanderbeck, F., Wolsey, L.A., 1996. An exact algorithm for IP column generation. Operations Research Letters 19, 151–159.

Warner, D.M., 1976. Scheduling nursing personnel according to nursing preferences: A mathematical programming approach. Operations Research 24, 842–856.

Weil, G., Heus, K., Francois, P., Poujade, M., 1995. Constraint programming for nurse scheduling. Engineering in Medicine and Biology Magazine, IEEE 14, 417–422.